Testing Metrics for Requirement Quality

Linda Rosenberg, Ph.D. [1]

Lawrence Hyatt [2]

Theodore Hammer [3]

Lenore Huffman [4]

William Wilson [5]

ABSTRACT

Key Words: Requirements, Metrics, Quality, Testing

1. Introduction

The National Aeronautics and Space Agency (NASA) is increasingly reliant on software for the functionality of the systems it develops and uses. The Agency has recognized the importance of improving the way it develops software, and has adopted a software strategic plan to guide the improvement process. At NASA's Goddard Space Flight Center (GSFC), the Software Assurance Technology Center (SATC) and the project Quality Assurance Office are working together to develop and apply a metrics program that utilizes the information available in the requirements phase of the software development life cycle. Metrics based on this information provides insight into the testing of requirements; this information assists the Quality Assurance Office in its project oversight role.

Requirements development and management have always been critical in the implementation of software systems—engineers are unable to build what analysts can't define. Recently, automated tools have become available to support requirements management. The use of these tools not only provides support in the definition and tracing of requirements, but it also opens the door to effective use of metrics in characterizing and assessing testing. Metrics are important because of the benefits associated with early detection and correction of problems with requirements. Problems that are not found until testing are at least 14 times more costly to fix than if the problem was found in the requirement phase.[2]

In this paper we will look at what constitutes requirement quality and the relationship of requirement quality to testing. We will discuss the quality attributes relevant for requirements and identify applicable metrics. The requirements segment of a metrics program is multifaceted—it evaluates the quality of the requirements document for quality indicators [9]; identifies volatility [4]; and tracks testing as a way to ensure that all requirements have been satisfied.[] These metrics assist project managers and quality assurance engineers in ensuring that the functionality specified by the requirements is contained in the completed software system. Data from NASA projects will be used in this paper to demonstrate the application and interpretation of these metrics. We will conclude with a brief discussion of how the requirement repository affects the metrics.

There are no published or industry standard guidelines for these metrics—intuitive interpretations, based on experience and supported by project feedback, are used in this paper. Project management has reacted favorably to these metrics and have used the analysis results to mitigate a perceived risk. The SATC continues working on methods to mathematically validate the intuitive guidelines. Joint work also continues to identify new metrics available through the application of a requirement management CASE tool. The objective is to assist project management in producing high-quality requirements and test plans.

2. Requirement Quality

Software requirements define the required functional requirements (what the software must do), performance requirements (how many and how fast) and interface requirements (with what, how, and with whom the software must interact). These requirements go through an iterative evolutionary process, clarifying concepts and adding additional levels of details at each iteration. If requirements are ambiguous, incomplete, or difficult to understand, the risk of an unsatisfactory final product is increased. Towards the end of the requirements phase, the requirements should stabilize with respect to additions, deletions and changes. All requirements should be traceable from their source to the software requirements document, through design and implementation, and then test.

The associated attributes for requirements quality are:

- Ambiguity - Requirements with potential multiple meanings.
- Completeness - Items left to be specified.
- Understandability - The readability of the document.
- Volatility - The rate and time within the life cycle changes are made to the requirements.
- Traceability - The traceability of the requirements upward to higher level documents and downward to code and tests.

The first facet of testing metrics in the requirement phase is a textual analysis of the requirement specification. Because both parties must understand requirements that the acquirer expects the provider to contractually satisfy, specifications are usually written in natural language. These requirements are then stored in a requirement database or repository. The use of natural language to prescribe complex, dynamic systems has at least two severe problems: ambiguity and inaccuracy. Many words and phrases have dual meanings which can be altered by the context in which they are used. Defining a large, multi-dimensional capability within the limitations imposed by the two dimensional structure of a document can obscure the relationships between individual groups of requirements. In addition to the language of the specification, the structure of the document reveals information about the organization, consistency and level of detail. These specification limitations and problems impact testability.

The second facet of metrics is the verification of the requirements, specifically the volatility and the traceability.

3 Requirements Specification

The SATC developed the Automated Requirements Measurement (ARM) tool to address management needs: that of providing metrics which NASA project managers can use to assess the quality of their requirements specification documents and that of identifying potential testability risks poorly specified requirements introduce into any project. The ARM tool searches the requirements document for terms the SATC has identified as quality indicators. Reports produced by the tool are used to identify specification statements and structural areas of the requirements document which need improvement. It must be emphasized that the tool does not assess correctness of the requirements specified; it does, however, assess the structure, language, and vocabulary of both the document itself and the individual requirements.

3.1 Textual Specification

There are aspects of the documentation that can be measured and therefore can be used as indicators of testability.

- LINES OF TEXT are the number of individual lines of text read by the ARM program from the source file. This data provides one measure of size that can be used to normalize the data and allow assessment guidelines to be developed for the other categories.

- IMPERATIVES are those words and phrases that command that something must be provided. "Shall" normally dictates the provision of a functional capability; "Must" or "must not" normally establishes performance requirements or constraints; "Will" normally indicates that something will be provided from outside the capability being specified. An explicit specification will have high counts in the report IMPERATIVE list (i.e. shall, must, required). These terms indicate testable functionality that needs to be verified in the final product.

- CONTINUANCES are phrases such as "the following:" that follow an imperative and precede the definition of lower level requirement specification. The extent that CONTINUANCES are used is an indication that requirements have been organized and structured. These characteristics contribute to the traceability and maintenance of the subject requirement specification. However, extensive use of continuances indicate multiple, complex requirements that may not be adequately factored into development resource and schedule estimates, and may require complex testing.

- WEAK PHRASES are clauses that are apt to cause uncertainty and leave room for multiple interpretations. Use of phrases such as "adequate" and "as appropriate" indicate that what is required is either defined elsewhere or worst, the requirement is open to subjective interpretation. Phrases such as "but not limited to" and "as a minimum" provide the basis for expanding requirements that have been identified or adding future requirements. WEAK PHRASE total is indication of the extent that the specification is ambiguous, incomplete and not easily testable.

- OPTIONS are those words that give the developer latitude in the implementation of the specification that contains them. This type of statement loosens the specification, reduces

the acquirer's control over the final product, and establishes a basis for possible cost and schedule risks. OPTIONS, since not required, may cause difficulty in testing.

Table 1 below contains the results from the analysis of 56 NASA requirement documents. The top row indicates the category and specific terms counted in that category. The last row is the analysis of the Detailed Requirement document for a very large project at NASA. (All projects are anonymous in publications.)

56 DOCUMENT	Lines of Text - Count of the physical lines of text	Imperatives - shall, must, will, should, is required to, are applicable, responsible for	Continuances - as follows, following, listed, inparticular, support	Directives - figure, table, for example, note:	Weak Phrases - adequate, as applicable, as appropriate, as a minimum, be able to, be capable, easy, effective, not limited to, if practical	Incomplete (TBD, TBS)	Options - can, may, optionally
Minimum	143	25	15	0	0	0	0
Median	2,265	382	183	21	37	7	27
Average	4,772	682	423	49	70	25	63
Maximum	28,459	3,896	118	224	4	32	130
Stdev	759	156	99	12	21	20	39
Project X	9,895	1,345	720	103	222	13	219

Table 1 : ARM Tool Data and Project Data

The project data is normalized based on size and then analyzed by the standard deviations from the average. Project X is about twice the size of the average document (based on Lines of Test), the number of imperatives is also almost twice the average so the amount of text is proportional to the number of requirement, not excessively wordy such that it would impede the writing of test cases. It has very few incompletes but a high number of weak phrases and optional phrases. These two problems, weak phrases and options, will make this document very had to test.

The raw data in Table 1 is informative but it is helpful to identify how project data relates to the average data using standard deviations. Figure 1 looks at this data using standard deviations.

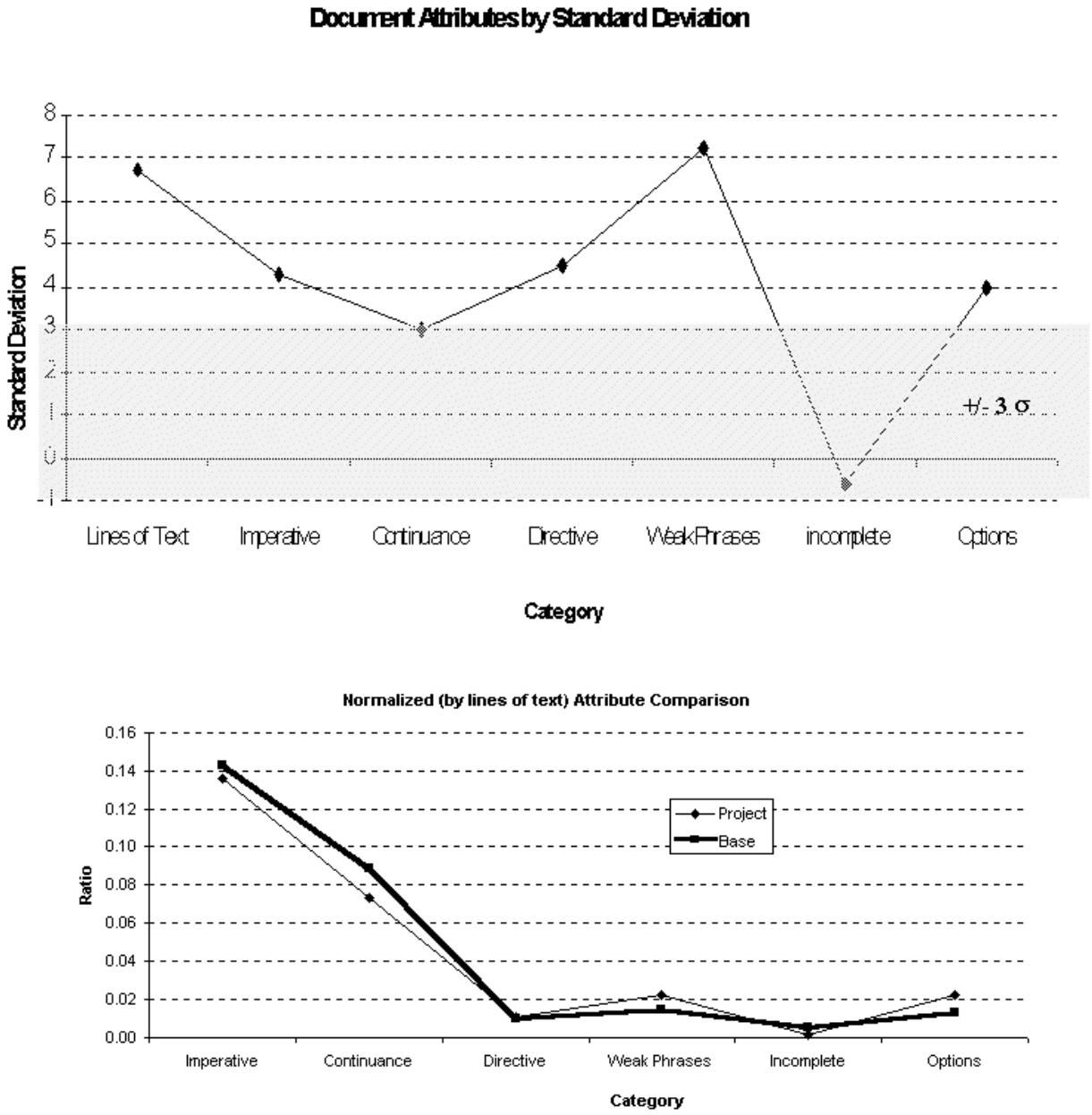

Figure 1: Attribute Comparison using Standard Deviations

For this project, again it is a large document in size, and has many pre imperatives, continuances, directives, weak phrases, and options. The number of incomplete phrases however, are few.

3.2 Requirement Structure

The structure of the document analyzed in Table 1 is also indicative of potential testing problems. ARM uses the structure depth and specification depth to depict two aspects of the document's structure.

- STRUCTURE DEPTH provides a count of the numbered statements at each level of the

source document. These counts provide an indication of the document's organization and consistency and level of detail. High level specifications will usually not have numbered statements below a structural depth of four. Detailed documents may have numbered statements down to a depth of nine. A document that is well organized and maintains a consistent level of detail will have a pyramidal shape (few numbered statements at level 1 and each lower level having more numbered statements than the level above it). Documents that have an hour-glass shape (many numbered statements at high levels, few at mid levels and many at lower levels) are usually those that contain a large amount of introductory and administrative information. Diamond shaped documents (a pyramid followed by decreasing statement counts at levels below the pyramid) indicate that subjects introduced at the higher levels are probably addressed at different levels of detail.

- SPECIFICATION DEPTH is a count of the number of imperatives at each level of the document. These numbers also include the count of lower level list items that are introduced at a higher level by an imperative that is followed by a continuance. This structure has the same implications as the numbering structure. However, it is significant because it reflects the structure of the requirements as opposed to that of the document. Differences between the shape of the numbering and specification structure are an indication of the amount and location of background and/or introductory information is included in the document. The ratio of total for SPECIFICATION STRUCTURE to total lines of text is an indication of how concise the document is in specifying requirements.

The application of the structure and specification depths is still under investigation, however, the initial results from Project X are interesting. Figure 1 depicts expected structure versus actual structure of the Specification and Design requirement documents.

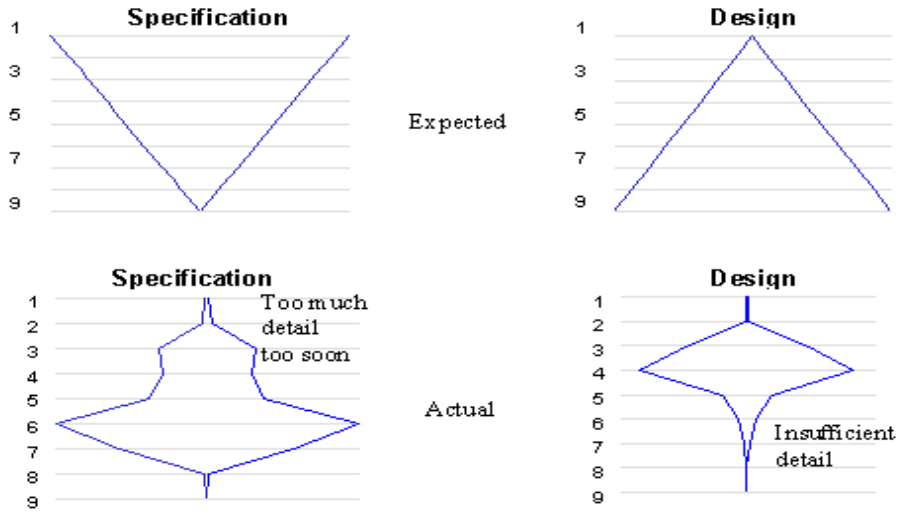

Figure 2 : Document Depth at Which Imperatives are Located

The project data indicates detail at low levels suggesting the Specification requirements may have been overly defined, therefore artificially constraining the design and its

expansion. The structure of the imperative levels in the Design document reinforces this observation, indicating little expansion was done where extensive expansion is expected and specification at high levels only.

4. Requirement Volatility

Industry has shown that unstable requirements increase the risks to final project.[] Developers cannot design or code a moving target and changes later in the life cycle have ripple effects that may impact both products and schedule. The earlier in the life cycle the requirements stabilize, the less changes to the test plans that are necessary. A volatile requirements document is one that is changed frequently. Since test cases should be completed by the end of the requirement phase [], volatile requirements also impact testability. Requirements that are changed after test plans are written can cause the wrong functionality to be tested. The first diagram in Figure 2 indicates the number of requirements is stabilizing and looks good but the second diagram indicates there are still numerous changes through the use of the "modified" category.

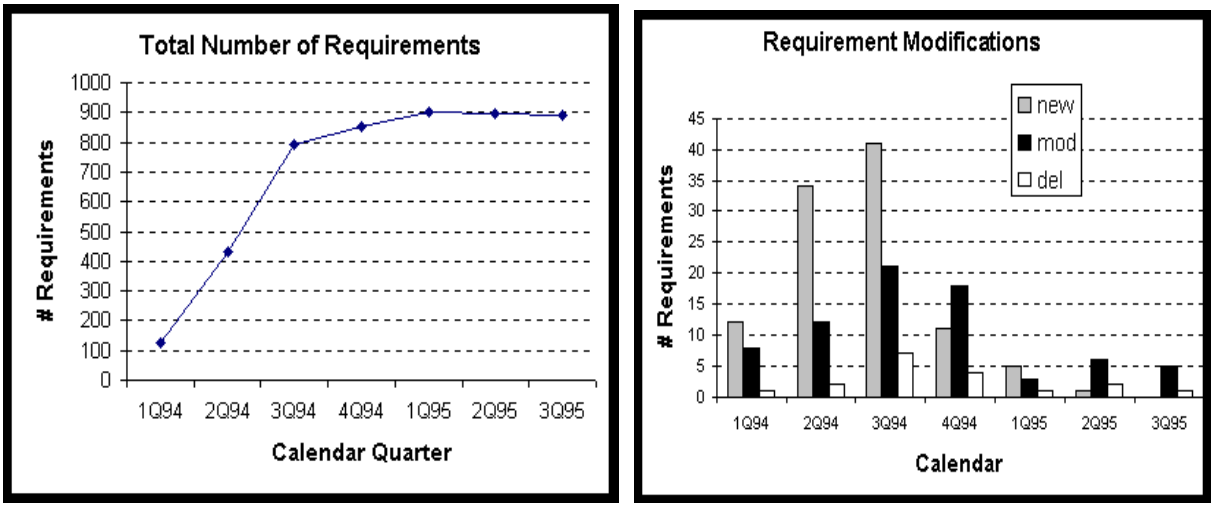

Figure 3 : Requirement Volatility - 2 Data Sets

Another example of requirements volatility is the shifting of requirements from one build to future builds. One of the purposes of a multiple build development effort is to minimize the implementation risk associated with any one build. This insures that no single build implements an inordinate number of requirements.[] Figure 3 shows the counts of the requirements for two builds of a NASA project. The impact to testing is that as the requirements shift from Build 1 to Build 2, testing must also shift. Since the number of requirements increased in Build 2, the resources to test must also be increased for Build 2or insufficient testing may result.

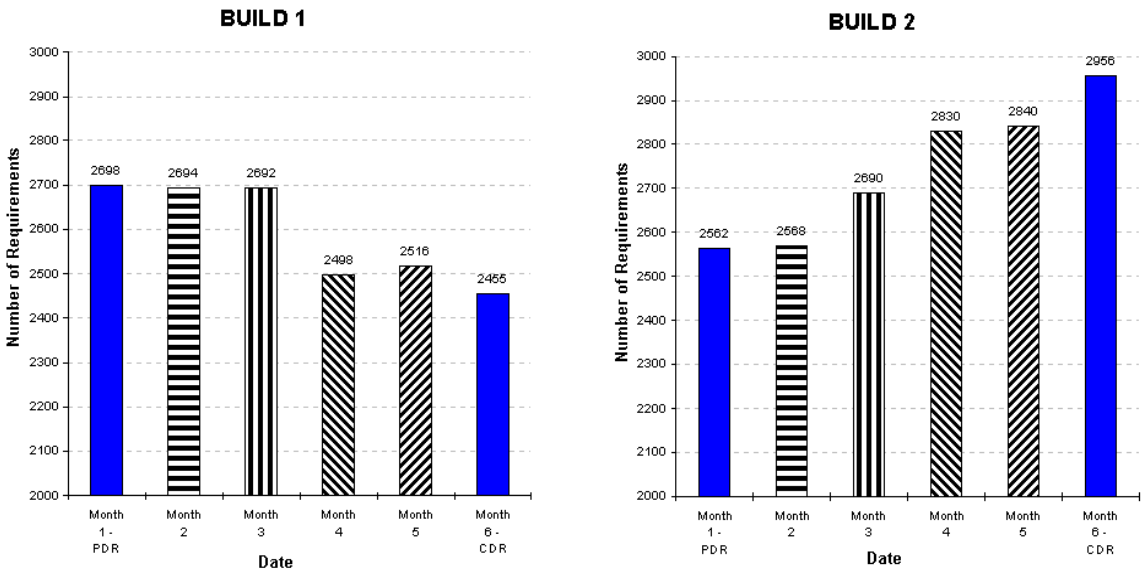

Figure 4 : Requirement Allocation by Build

4 REQUIREMENT TESTING

Once requirements are written, methods for ensuring that the system contains the functionality specified must be developed. There are a number of methods for validating functionality and determining if the software reacts in the expected way: testing, inspections, analysis, and demonstrations.[2,8] The main method used in all systems is testing. To validate the requirements, test plans are written that contain multiple test cases; each test case is based on one system state and tests some functions that is based on a related set of requirements.[6]

4.1 Requirement Links to Tests

The first objective is to verify that each requirement will be tested; the implication is that if the software passes the test, the requirement's functionality is included in the system.[1] This is done by determining that each requirement is linked to at least one test case. It is expected that each requirement will be linked to multiple test cases, and that each test case will test multiple requirements.[2,6,8] These relationships are shown in Figure 4.

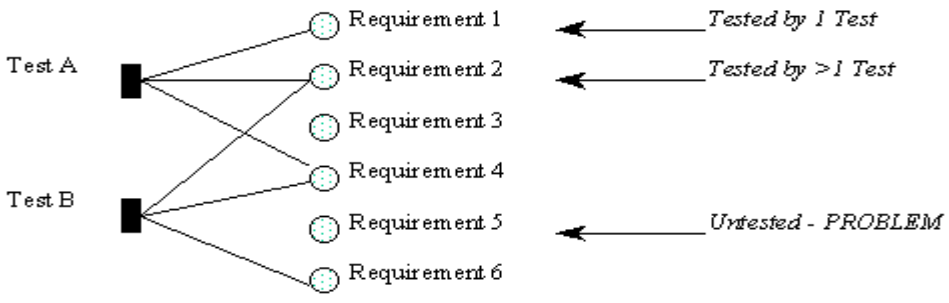

Figure 5 : Example Test Linkage

Figure 5 is an application of this data for a NASA project with 3 builds. Build 1 appears greater than 60% unlinked due to database problems, the database was not created until after Build 1, hence most of the data from Build 1 was not entered. Build 2 is currently testing Detail requirements, but 40% of the requirements are not linked to any test. This data indicates that there is no way to verify whether the functionality of approximately 1,000 requirements is included in the system. It may be possible to link some of the requirements to existing test cases with minimum modification to the test data. If new test cases must be developed, budgetary problems may be created and the testing schedule must be increased. In all cases, further investigation of the missing links is warranted. For Build 3, just starting the coding phase (coding continues for approximately 10 additional months), only 25% of the requirements are not linked to test cases. This situation needs to be monitored on a monthly basis but is not one for major concern at this time.

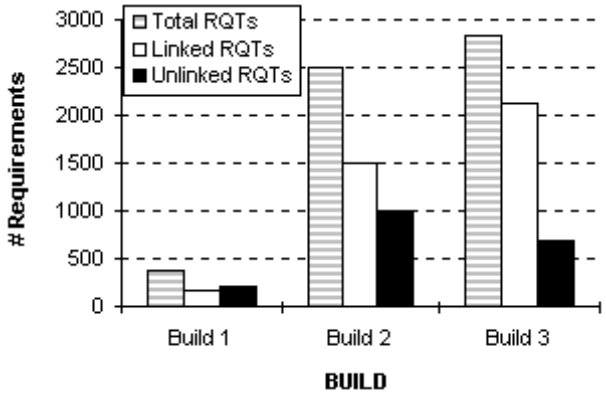

Figure 6: Requirement Linkage to Tests

[1] **SATC Unisys GSFS NASA, 301-286-0087, Linda.Rosenberg@gsfc.nasa.gov**

[2] **GSFC NASA (retired)**

[3] **GSFC NASA, 301-286-7475, Ted.Hammer@gsfc.nasa.gov**

[4] **SATC Unisys GSFS NASA, 301-286-0099, Lenore.Huffman@gsfc.nasa.gov**

[5] **SATC GSFC NASA (retired)**

Testing Metrics for Requirement Quality was presented at the 11th International Software Quality Week, May 1998 in California.